MOSAIC User Guide

Stefan Kuntsche

October 17, 2011

d|b|t|a

Department of Process Dynamics and Operation

Contents

1 Introduction **3**

2 Modeling Concepts **5**

2.1 Motivation 5

2.2 Modeling Elements in Detail 7

2.2.1 Notations and Variable Namings 7

2.2.2 Equations 7

2.2.3 Equation Systems and Variables 7

2.2.4 Functions 8

2.2.5 Connectors 8

2.2.6 Interfaces 8

2.2.7 Units, Ports, and Streams 9

2.2.8 Parameter Lists 9

2.2.9 Evaluations 9

2.2.10 Variable Specification Lists 9

2.3 Connection Techniques 10

2.3.1 Naming Policy 'Integrated' . . . 12
2.3.2 Naming Policy 'Encapsulated' . . . 16
2.3.3 Using External Ports . . . 19
2.3.4 Using Internal Streams . . . 21

3 Mathematical Expressions in MOSAIC 26
3.1 Conventions for Mathematical Expressions . . . 26
3.2 Latex Standard for MOSAIC . . . 29
3.3 The correct use of indices . . . 31
3.3.1 Index specifications in Equations . . . 31
3.3.2 Index specifications in Functions . . . 31
3.3.3 Index specifications in Connectors . . . 32
3.3.4 Index specifications in Interfaces . . . 32
3.3.5 Index specifications in Parameter Lists . . . 32
3.3.6 Index specifications in Summation expressions . . . 32
3.3.7 Index specifications in Evaluation::Indexing Panel . . . 32
3.4 Index values as Variables . . . 33

4 Standard Components of the User Interface 34
4.1 File Selection . . . 34
4.2 Tables . . . 34
4.3 Math Editors . . . 35
4.4 Modeling assistance . . . 36

5 Model parts and corresponding editors 36
5.1 Notation Editor . . . 37
5.2 Equation Editor . . . 38
5.3 Function Editor . . . 39
5.4 EQSystem Editor . . . 39
5.4.1 Connected Elements Editor . . . 40
5.4.2 Internal Streams Editor . . . 40
5.4.3 External Ports Editor . . . 41
5.4.4 Function Usage Editor . . . 42
5.5 Evaluations . . . 43
5.5.1 Equation System . . . 43
5.5.2 Indexing . . . 44
5.5.3 Instance . . . 44
5.5.4 Info . . . 44
5.5.5 Variable Specification . . . 45
5.5.6 Parameter Specification . . . 45
5.5.7 Evaluation . . . 46
5.5.8 Results . . . 46
5.6 Interfaces . . . 47
5.7 Connectors . . . 47

5.8 Parameter Lists 48

6 Tutorials **49**
6.1 Getting Started - Create and solve a very simple model . . . 49
6.2 Notations and Variables - Superscripts, Subscripts, and Indices 52
6.3 Differential Equation Systems 58
6.3.1 ODE - The Van der Pol Oscillator 58
6.3.2 DAE - The Robertson Problem 60

7 Appendix **63**
7.1 Connection using streams 63
7.1.1 Objects 63
7.1.2 Name Spaces 64

1 Introduction

The modeling environment MOSAIC is a project that tries to improve the creation and the use of custom made models. While many software tools exist that provide standard models for chemical engineering (like AspenPlus among many others) it is often necessary to create new models that fulfill special needs. Such custom models can be written in many languages (e.g. Fortran, C) and there are many environments that allow use custom models in one or two languages (e.g. Matlab, gProms). However, most of these languages are textual programming languages, which leads to three handicaps: First, it takes a long time to write and debug the model; second, the model can only be used in environments that understand the programming language of the model, for other environments the model must be programmed anew which again comes along which much effort; and third, there is a visual and conceptional gab between programming languages and the mathematical two-dimensional formulation. To react on this situation, MOSAIC is created as symbolic mathematical environment and code generator for large systems of equations. The idea is to create a model once and use it in different cases and many environments or languages.

Apart from its modeling concept, MOSAIC is a Web 2.0 application, so that the maintenance on the computer of the user is small and the storage is done on a server over the internet.

Important principles To describe the work with the MOSAIC modeling environment the following basic principles need to be mentioned:

- Mathematic expressions are written symbolically so that they can be seen in the environment just as they would appear in a publication or written on paper.

- Equations and functions are created separately and then assembled to equation systems.
- In the same way equation systems can be added to other equation systems.
- Every equation, function and equation system must have a notation that contains an explanation for each symbol used in the mathematical expressions.
- The calculation of an equation system is separated from its creation. This makes the equation systems more reusable. In particular
  - an equation system does not contain any specific values for its variables and parameters
  - an equation system does not contain any information about which are the design variables and which are the iteration variables.
  - an equation system does not contain information about the maximum values of indices.
  - such pieces of information are kept in an object called evaluation.

General modeling procedure To build up a new model from scratch you have to adhere to the following procedure.

- Create a notation object containing information for all symbols that you will use.
- Create the equations and assemble them to an equation system. (If necessary, you can sub-divide your model into several equation systems, which can be added to a superior equation system in a second step.)
- If you need functional calculations you can create function objects. Instances of function objects can be added into the equation system.
- Once you have created your equation system, create an evaluation object that contains all information necessary to specify a problem to be solved based on the equation system. This step includes
  - Specification of maximum index values (e. g. number of components)
  - Classification of the variables into design and iteration variables.
  - Giving values to the design variables and parameters, as well as, providing guess values for the iteration variables.

- With all necessary information gathered, create a problem solving program code in one of the languages provided by MOSAIC. Depending on what code generator you used you can either
  - compile and execute the code on the solver which allows you to see the results directly in the MOSAIC user interface
  - copy the code and use it in your own modeling environment.

Re-use Usually you do not want to build up everything from scratch. Therefore all objects created in MOSAIC can be used in several models. This allows you for example to build up your notation suitable for the current project and use it in all related models. In the same way you can reuse equations and entire equation systems. There are several ways to add existing equations or equation systems to a new equation system. To be used together two equation systems do not need to use the same notation. When existing equations and equation systems are integrated in a new equation system the meanings of the variables of all connected model parts must be clear throughout the equation system to be evaluated. In MOSAIC this issue is addressed by giving synonyms to variables whenever variables of the same meaning have different names. This will be explained in greater length later in the manual.

2 Modeling Concepts

2.1 Motivation

Proximity to literature In chemical engineering a large part of the models is based on conservation laws of mass, impulse and energy, transport equations, summation relations, and further phenomenon-specific relations. The resulting MESH model results in an equation system which is, depending on the special case, a root finding problem, an ordinary differential or differential algebraic equation system. Such equation systems are also a very important means of description in other engineering disciplines. The presentations of such models in literature consist of a list of the relevant equations given in mathematical expressions using symbols that are specified in a given notation. The notation adds information to the model that is necessary for the correct use of the equations. The information given for each symbol is an important part of documentation for the creator of the model equations as well as for fellow researchers, students, and programmers. In literature, one variable can be named by several symbols including subscripted and superscripted elements; as opposed to an outright sequence of characters in conventional programming using languages like Fortran or Matlab. To reflect mathematical expressions in MOSAIC, the variables are distinguished by their Variable Naming which consists of symbols on the

base line and may have several superscripts, one fixed subscript and several indices taking values from 1 to a given maximum value. Furthermore, the Notation is a centrally important model element in MOSAIC that reflects this classification of naming compartments.

Modularity Writing equations into a document or into a solving software takes time and is an error-prone process. In such a process many equations are rewritten many times: if a new model is created, the engineer often has to implement equations he has already written and corrected in other models. Furthermore, many equations are reimplemented the same way independently by different engineers. If sections of the model are re-written, time and personal resources are consumed, that could be used in much better ways. It makes sense to be able to re-use equations and equation systems that have been well tested. Thus, the user can create the equations that are essential in his model and that are really new and different from existing models, and he can use ready made equations for standard model parts. The reusability of small model parts down to equations is reflected in MOSAIC. This is done by establishing the Equation as an independent model element. The model element Equation System is a combination of several Equation elements. Several Equation System elements can be put together to create a new Equation System.

When one equation system is added to another equation system with the same notation the coupling of the equations is directly visible by the names of the variables involved. In literature as well as in computer programs, however, the notations differ from model to model, so that variables with the same physical meaning may have different names. In general, different models will also use a different set of units. On the other hand, it might be desirable to see all model parts in one and the same notation even if some of them were created using a different notation. In this case, the variable would have at least two names: one in the original notation and one corresponding to the notation in the superordinate model. Further names may need to be added in the same way if the superordinate model is added to other models. To reflect this, MOSAIC distinguishes between Variable and Variable Naming. One Variable has one or several Variable Namings.

As the size of the equation systems grows, it becomes more and more difficult to provide distinct names in one notation. For example the symbol y_i as vapor molar fraction for component i may be used in many Equation Systems that represent process units. Furthermore, this variable might occur in sub-models, etc. To avoid confusion and difficult naming conventions of variables, MOSAIC allows the assignment of Equations and Equation Systems to different name spaces. Thus, equivalent Variable Namings assigned to different name spaces are considered as belonging to different variables. How to combine models in such a way is described in more detail in section

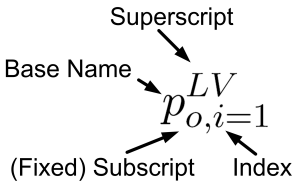
2.3 Connection Techniques.

2.2 Modeling Elements in Detail

2.2.1 Notations and Variable Namings

Notations are the fundamental modeling element in MOSAIC. They are used to give the symbols appearing in the equation systems a meaning that will be visible to the user throughout the entire modeling process. Notations in MOSAIC are a direct projection of notations found in the literature: The notation belongs to a mathematical model. All symbols are listed systematically. Each symbol is followed by a short description, which provides its meaning in the context of the mathematical model that both the symbol and the notation belong to. As it is good practice in the literature, the Notation in MOSAIC divides the symbols into groups according to their position and their functionality. The following four groups are used in MOSAIC

- Base Names - symbols that appear on the base line of the written formula
- Superscripts - raised symbols
- Subscripts - lowered symbols as fixed descriptive elements
- Indices - lowered symbols that take a value from 1 to a specifiable maximum value.



2.2.2 Equations

The Equation is the simplest and smallest element in MOSAIC. It must use a Notation, it must have a Description, and it has a MathML expression containing the mathematical content. It may also use a Parameter List to point out variables that should be treated as global parameters.

In the current version of MOSAIC, a Latex subset is used to create the MathML expressions.

2.2.3 Equation Systems and Variables

Equation Systems are the modeling object that combines the information of the mathematical content. They are created by combining equations, functions, and other equation systems. Furthermore, the use of External Ports

and Internal Streams as a projection of process unit flow sheeting is done within the Equation System. On the other hand, Equation Systems can be evaluated by specifying the design variables and other problem specific information (see **2.2.9 Evaluations**. Thus, they are the center piece between the modular pieces of information in equations and concrete modeling tasks that give you the information you need.

2.2.4 Functions

Functions are very useful when values need to be calculated and it is undesirable to introduce additional variables or equations. Possible applications include calculation of enthalpy, phase equilibrium factors, etc. Functions in MOSAIC have one output value and several input values. It is possible to provide both the output value and the input values with an index.

Appliances In general, the Notation of a Function is independent from the Notation used in the Equation System. There must be a mechanism to define which of the variables in the Equation System are the input values and which is the output value. In programming languages the input values are assigned according to the order in which they appear within the function call. The modeling in MOSAIC, however, is focused on the Variable Naming and pursues the concept of synonymous Variable Namings. Thus, the function is applied by assigning the Variable Namings of the input variables and the output variable of the Function explicitly to the namings of the corresponding variables in the Equation System. The advantage of applying the Function in this way is that the focus is kept on the physical meaning of the variable. Errors from handing over variables in the wrong order are thus eliminated.

2.2.5 Connectors

Connectors are modeling objects that translate between two different notations. To make an equation system or equation using notation 'A' usable in an equation system using another notation ('B') you need to specify a connector that provides all pertinent variables with a synonym complying to notation 'B'.

2.2.6 Interfaces

Interfaces provide a norm for the shared use of variables. In principal, interfaces represent an independent list of variable namings that are either allowed or expected to be shared with another modeling element. They are used in ports and streams (explained later) to further extend the re-usability of existing units.

2.2.7 Units, Ports, and Streams

Units Large Equation Systems may represent a unit in the chemical engineering sense. It makes sense to restrict the accessibility of the variables of such Equation Systems and in a second step add standardized output interfaces to them. In this way, all incoming and outgoing material streams in a model of a process unit could have exactly the same Notation. But it would also be possible to define interfaces that represent control input and output, etc. To provide this kinds of standardized access points, MOSAIC allows the user to specify ports. Two units that have ports can be connected by streams.

Ports belong to an Equation System and cannot exist independently. They have a distinct name and an indication of the Interface they use.

Streams A stream just as a port cannot exist independently from an Equation System. It can combine exactly two ports. It has a distinct identification number, the names of the two ports it connects and an indication to the interface it uses.

2.2.8 Parameter Lists

Parameter Lists allow for a separation of certain variables within the equation system and treat them as global parameters. One useful application is the specification of parameters in functions for physical properties. For variables in parameter lists MOSAIC does not try to find synonyms. In the degree of freedom calculation they are automatically considered as a special form of design variable and are kept in a separate list.

2.2.9 Evaluations

Evaluations are the model element where a simulation problem is formulated based on an equation system. Here you specify the maximum values for the indices, choose the design variables, specify values for the design variables and guess values for the iteration variables. It is also necessary to generate program code to solve the problem. It is possible to choose between several code generators. Some of them provide code that is executable on the server. If one of these code generators it chosen, the user can also solve the problem and view the results directly from the web browser.

2.2.10 Variable Specification Lists

An important step in evaluating an Equation System is to specify which are the design variables and to give values for design variables as well as initial values for iteration variables. A good choice of initial values is as important

as a sensible selection of design variables. To enter all the values takes time and therefore this piece of information is stored in a separate modeling element. Variable Specification Lists can be stored on the user's MOSAIC account on the server as well as downloaded to their local computer. Variable Specification Lists can be opened and edited with MS Excel. There, the modeler can make use of functionalities like sorting lists, adding sequences of values, and using functions and graphs to generate and adjust sequences of values.

2.3 Connection Techniques

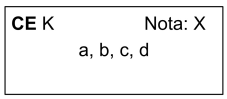
To introduce the different possible ways of connecting Equations and Equation Systems some general facts are listed below.

- Equation Systems are put together by 'connection'.
- An Equation System has one or more Connected Elements.
- A Connected Element is
  - an Equation or
  - an Equation System.

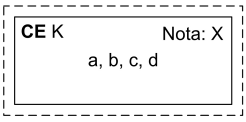
  Equations and Equation Systems can be connected to an Equation System, which can be connected to another Equation System and so forth. Therefore both Equations and Equation Systems are referred to as Connected Elements.
- 'Connection' means:
  - One or several subordinate Connected Elements are linked to a superordinate Connected Element.
  - The subordinate Connected Elements contain new information that is being added to the superordinate Connected Element.
- In one connection,
  - all Equations of the subordinate Connected Element are added to the superordinate Connected Element,
  - all Variables of the subordinate Connected Element are registered in the superordinate Connected Element. In the process of this registration variables of the same meaning are matched and thus considered as one variable once the connection process is finished.
  - Which variables have the same meaning can be specified in several ways. The basic mechanism in this process is the comparison of the Variable Namings (see 2.2.1 Notations and Variable Namings)

- Use of 'super' and 'sub':
  - The superordinate Connected Elements and all related model elements are identified by the word 'super', e.g. Super Connected Element, Super Equation System, Super Notation, Super Variable Naming, etc.
  - Accordingly, the subordinate Connected Elements and all related model elements are referred to e.g. as Sub Connected Element, Sub Equation System, Sub Notation, etc.

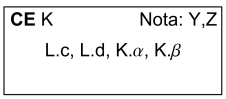
Symbols used for explanation In the following section symbolic charts are used to explain the different ways of connecting equations and equation systems. Below is an explanation for the symbols used, to make the charts understandable.



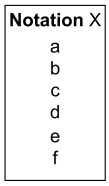
Equation or Equation System named K, containing the identifiers a, b, c, and d, while using the notation X. Variable names are matched with those found in implementing super Equation Systems .



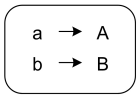
Equation or Equation System named K, containing the identifiers a, b, c, and d, while using the notation X. Upon connection to a superior equation system, variables given in a connector are translated and then matched. The other variables are not translated and given a separate name space.



Name spaces: Equation system named K, where variables from different name spaces (L and K) and from different notations (Y and Z) are present. In this Equation System, variables c and d belong to name space L, variables α and β belong to name space K.

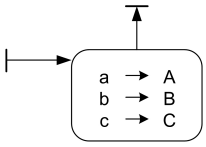


Notation named X containing descriptions for the identifiers a, b, c, d, e, and f.

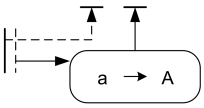


Connector translating variable names a to A and b to B where a and b are described by the Sub Notation and A and B by the Super Notation.

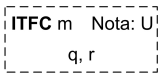
Connection, the arrow points either from the Sub Equation System to the Super Equation System.



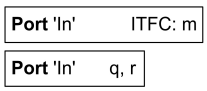
Connection using a Connector where all variables of the Sub Equation System have to be translated.



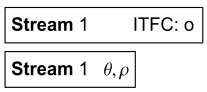
Connection using a Connector where only the variables named in the Connector are translated. All the other variables are only registered and given a separate name space.



Interface named m, using notation U and containing variables q and r.



Port with name ID 'In', using interface m. The lower box represents the abbreviated symbol for the Port, which directly shows the variables of the Interface.



Stream with ID number 1, using interface o. The lower box represents the abbreviated symbol for the Stream, which directly shows the variables of the Interface.

2.3.1 Naming Policy 'Integrated'

This way to connect Equations and Equation Systems is useful to build up Equation Systems of one Notation or to integrate Equations that use other Notations into the naming concept of the user's own project.

Main rules

- One Super Notation describes all Variable Namings.
- If different Notations are used, all Variables with a Variable Naming from subordinate Notations must be translated.

Main behavior and consequences

- In the superior Equation System Variables are distinguished only with respect to their name in the Super Notation.
- A Variable's Connected Element of origin is not used to distinguish it from other Variables.

Useful applications

- Build up a model from scratch using one well defined Notation.
- Integrate small model parts from other authors.
- Build up a library using a well defined Notation

Main advantage All Variables are viewed and documented according to the Super Notation (only one Notation)

Main disadvantage When working with several notations and big sub equation systems, it can be very hard to translate all variables of the sub system.

Example one All Connected Elements use the same Notation. The variables are matched according to the name they have in the subordinate Connected Element. This example is illustrated in figure 1.

Example two One subordinate Connected Element uses a different Notation than the superordinate Connected Element as is shown in figure 2.

Example three Two sub ordinary Connected Elements use different notations. They are connected to a Connected Element using still another notation. All variables have to be translated accordingly. In the result all variables have names according to the notation of the superordinate Connected Element, as in figure 3.

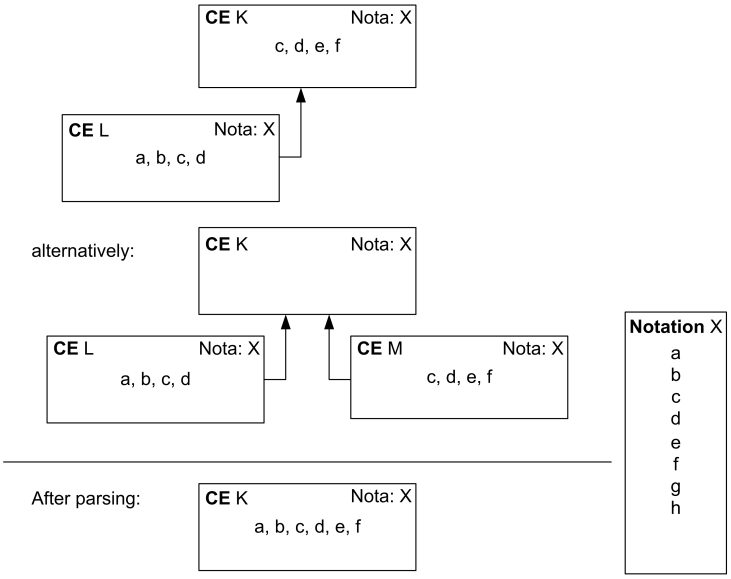


Figure 1: Naming policy 'Integrated' with the same Notation in all elements. **Top:** Connection of Sub Connected Element L to Super Connected Element K that contains variables prior to connection, **Middle:** Connected Elements L and M connected to initially empty Connected Element K, **Bottom:** resulting Connected Element

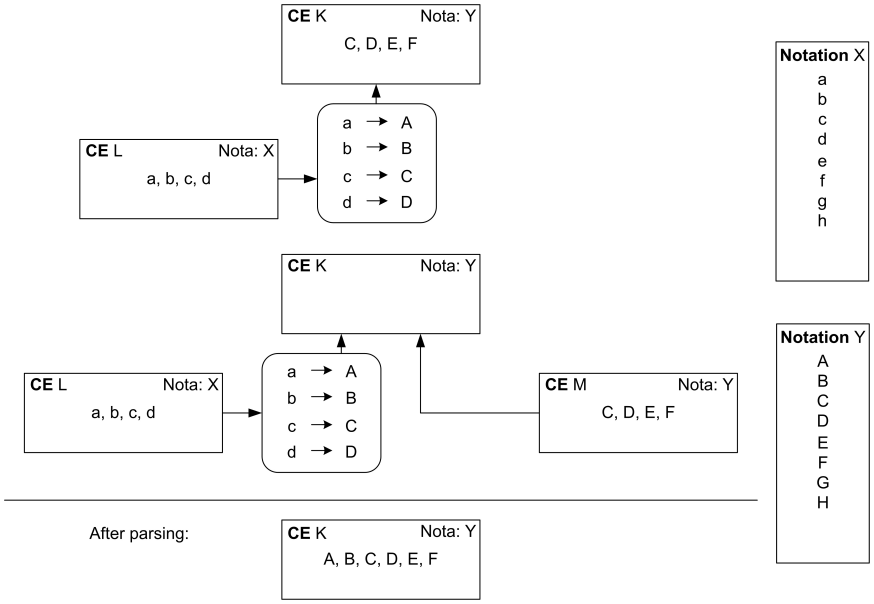


Figure 2: Naming policy 'Integrated' with different Notations. **Top:** Connection to a Super Connected Element that contains variables, **Middle:** Connection to empty Connected Element, **Bottom:** resulting Connected Element

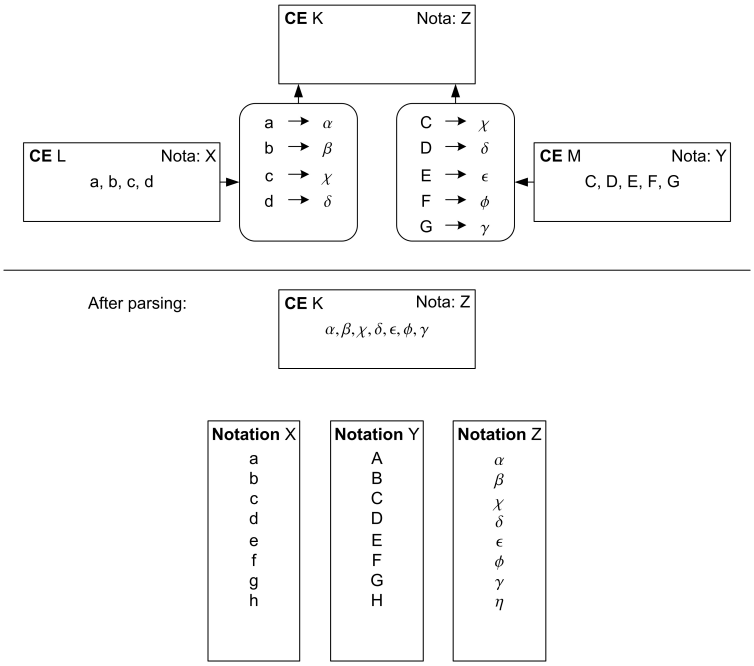


Figure 3: Naming policy 'integrated' with different Notations in all of the involved Connected Elements

2.3.2 Naming Policy 'Encapsulated'

This way to connect Equations and Equation Systems is useful to build up Equation Systems with different Notations when it is undesired to translate all Variables to one naming concept.

Main rules

- Only Variable Namings specified in a Connector are translated and integrated into the Super Notation.
- All variable names of the Sub Connected Element that are *not* translated are kept separate from those of the Super Connected Element.
- The separation is done by name spaces.

Main behavior and consequences

- Super Notation describes at least the Variable Namings that have been used as Super Variable Namings in the Connector.
- Encapsulated Sub Equation Systems keep their structure.

- Naming conflicts between encapsulated Sub Equation Systems and Super Equation Systems are avoided.
- Coupling of Equation Systems is done selectively by Connectors.

Useful application

- Add large and complex Equation Systems to the user's own Equation System (e. g. assemble a plant equation system from unit equation systems created by different groups and using different notations).

Main advantage The focus is given to the modularity of the Equation System. Reuse of well tested modules is made easy.

Main disadvantage Variable Namings are not translated between the Sub and Super Notations. Although they will be distinct in a mathematical way. Thus, their display might lead to confusion on first sight.

Example one One Sub Connected Element uses a different Notation and is connected using the policy 'Encapsulate', as in figure 4.

Example two Two Sub Connected Elements use different Notations. They are connected to a Connected Element using still another notation. The names of the variables that have the same physical meaning have to be translated into the Super Connected Element's Notation, as in figure 5.

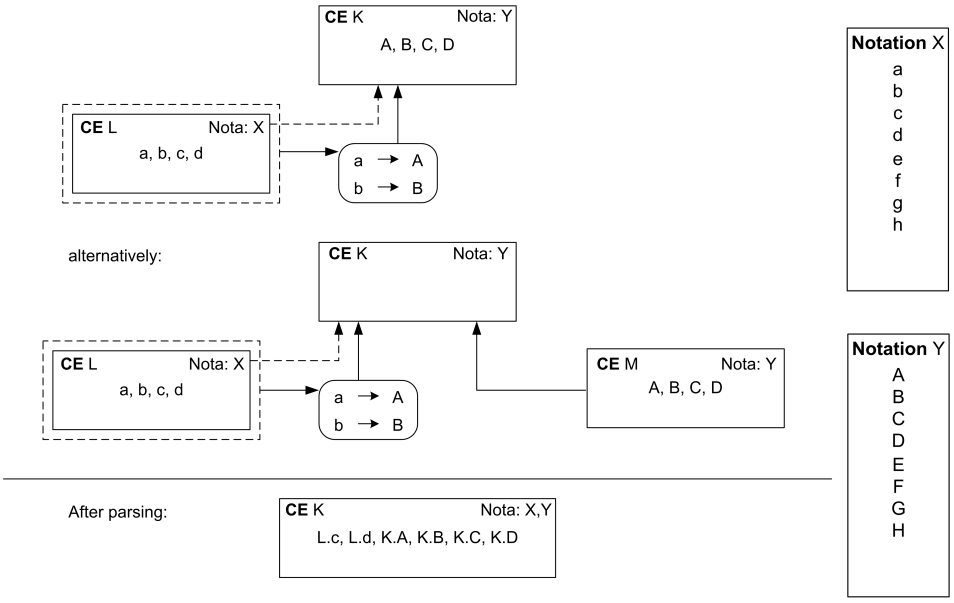


Figure 4: Naming policy 'encapsulate' with different Notations. **Top:** Connection to a Connected Element containing variables, **Middle:** Connecting two elements into an empty Connected Element, **Bottom:** resulting element.

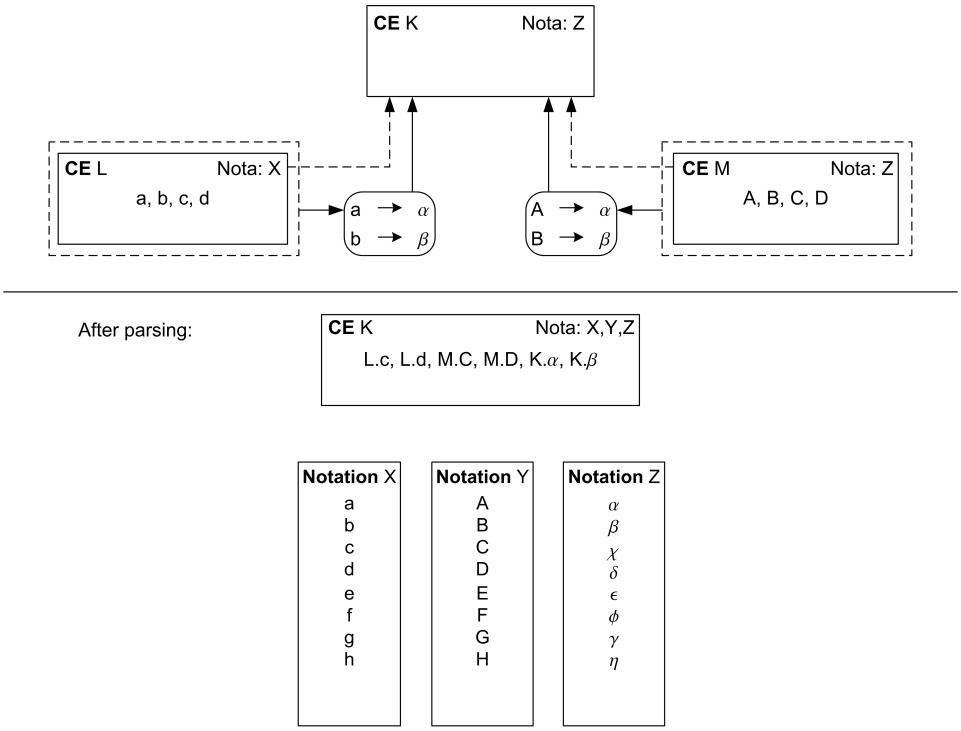


Figure 5: Naming policy 'encapsulate' with different Notations in all of the involved Connected Elements

2.3.3 Using External Ports

External ports are useful if standardized input or output is desired.

Main rules

- A port contains an Interface that specifies the allowed Variable Namings, and a unique name to distinguish it from the other ports.
- All variables that are projected in a port are given a new Variable Naming according to the port's Interface. This naming includes a distinct Name Space for the port. (A Connector is necessary for this projection). Variables not projected in a port keep their Variable Naming including the Name Space.

Main behavior and consequences

- Equation Systems are provided with customizable interfaces.
- The original Equation System is not changed in any way.

- One Equation System can be used as the basis for various independent port configurations.

Useful applications

- Equation Systems that represent units in chemical engineering can be used with standard inputs and outputs.
- Prepare Units to work together in Flow Sheets
- Provide the necessary port specification for Code Generation in commercial Flow Sheeting software.

Main advantages

- This feature offers a standardization that is highly customizable as the Interface used can be created by the user.
- Although standardized interfaces for data exchange are provided, the original Equation System is not changed.

Main disadvantage The user must specify a large amount of translation information in Connectors.

Example An Equation System is provided with two ports.

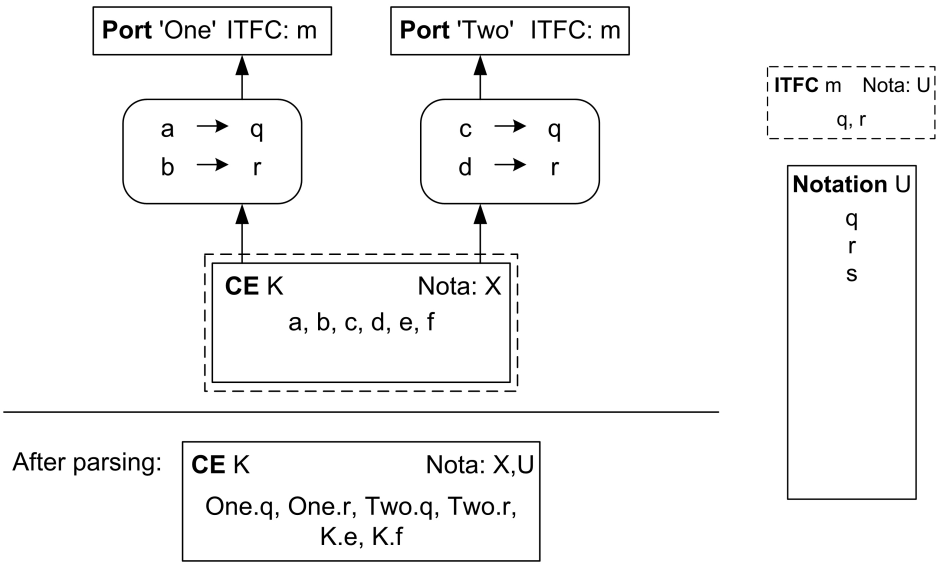


Figure 6: Equation System using ports

For the explanations in the following section, an abbreviated version of the figure 2.3.3 is given:

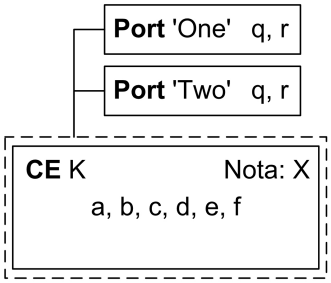


Figure 7: Abbreviated symbolic for an Equation System using ports

2.3.4 Using Internal Streams

Internal streams are useful to connect equation systems that provide external ports.

Main rules

- A Stream connects two Ports.
- A Stream has an ID-Number that is unique in the Equation System. Further, it has an Interface that specifies the allowed Variable Namings, and it contains the two names of the Ports it connects.
- The variables in the connected Ports are projected in the Stream. That means that their Variable Namings are translated to namings that are compliant to the Stream's Interface. (A Connector is necessary for this translation).

Main behavior and consequences

- All variables that have been projected into a Stream have standardized names.
- These variables can be seen easily when examining the model.

Useful applications

- Put together Equation Systems that use External Ports so that they can be used for flow sheeting.
- As the Interfaces can be created by the user, the data structure Stream can be used to model material streams of different information content as well as control flows and similar modeling tasks.

Main advantage Free definition of Stream variables possible.

Main disadvantage The user must specify a large amount of translation information in Connectors.

Example Two independent research teams assemble existing Equation Systems and provide them with Ports. In this step both research teams use their own Notations and their own Interfaces. After that, a third research team uses the models of the two former teams to build a process connecting both models by Streams. The third research team also uses its own independent Interface. Figures 8 through 12 illustrate that kind of connection.

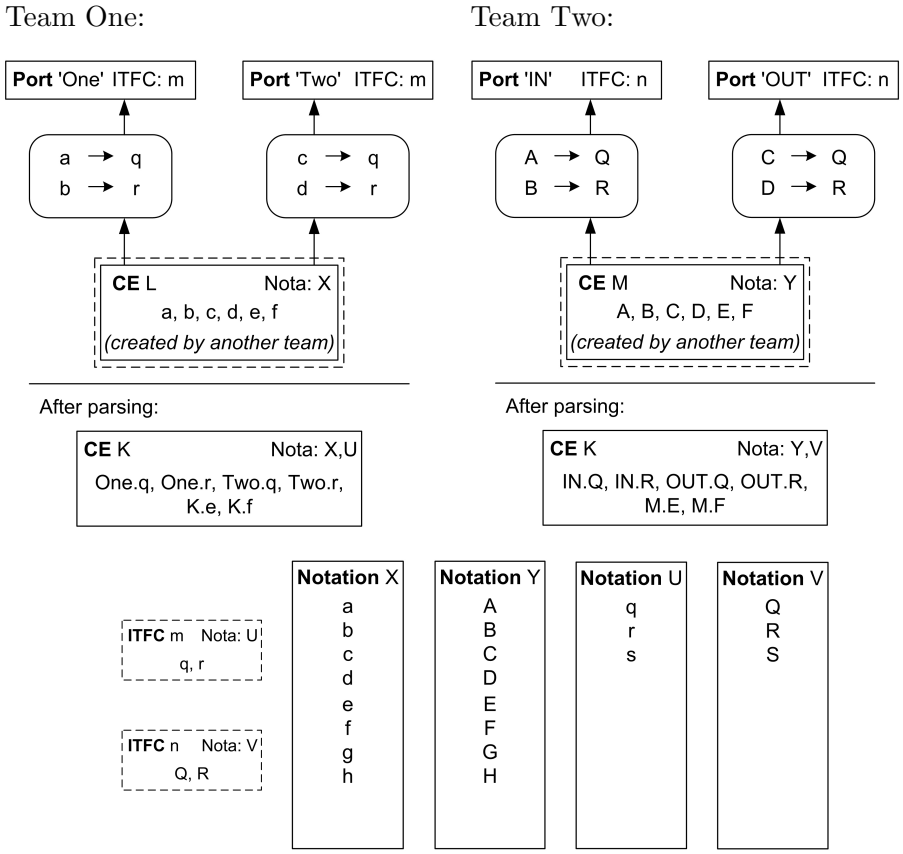


Figure 8: Units using ports created by two different teams.

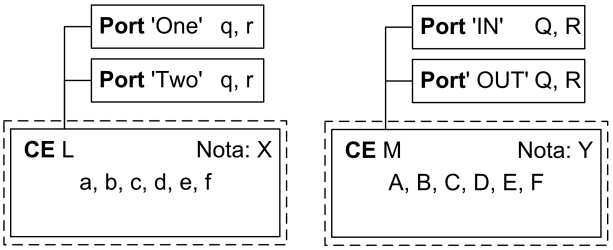


Figure 9: Abbreviated images for both Equation Systems of this example

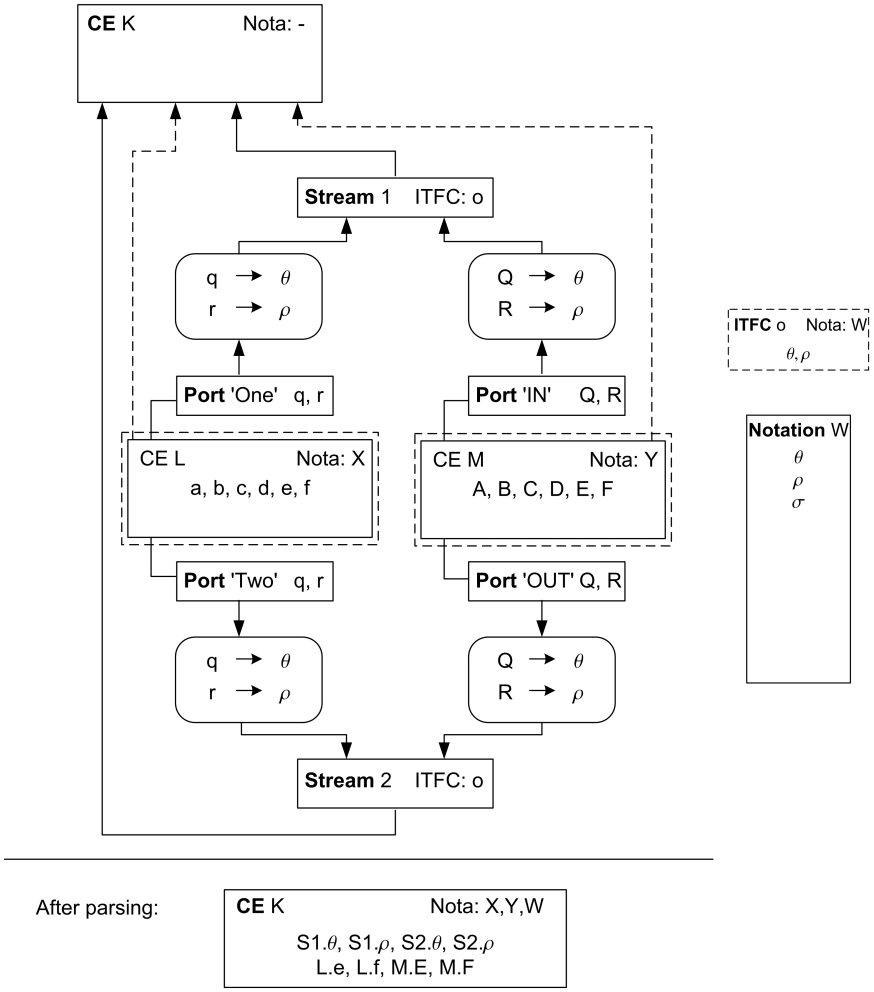


Figure 10: Connection via streams of two equation systems that have ports. The general case.

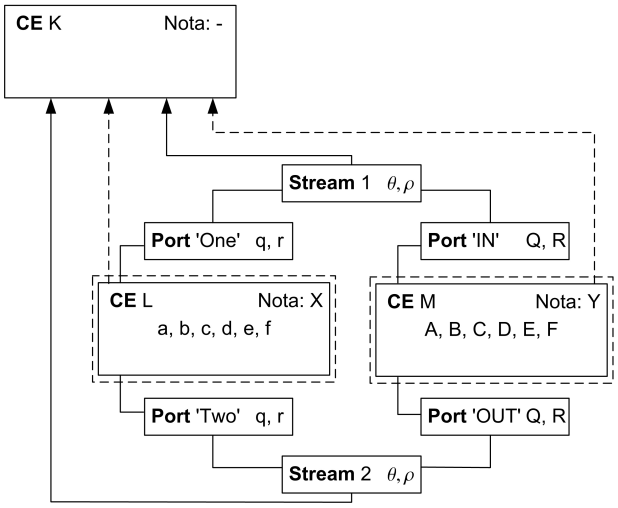


Figure 11: Abbreviated symbolic for a connection via streams

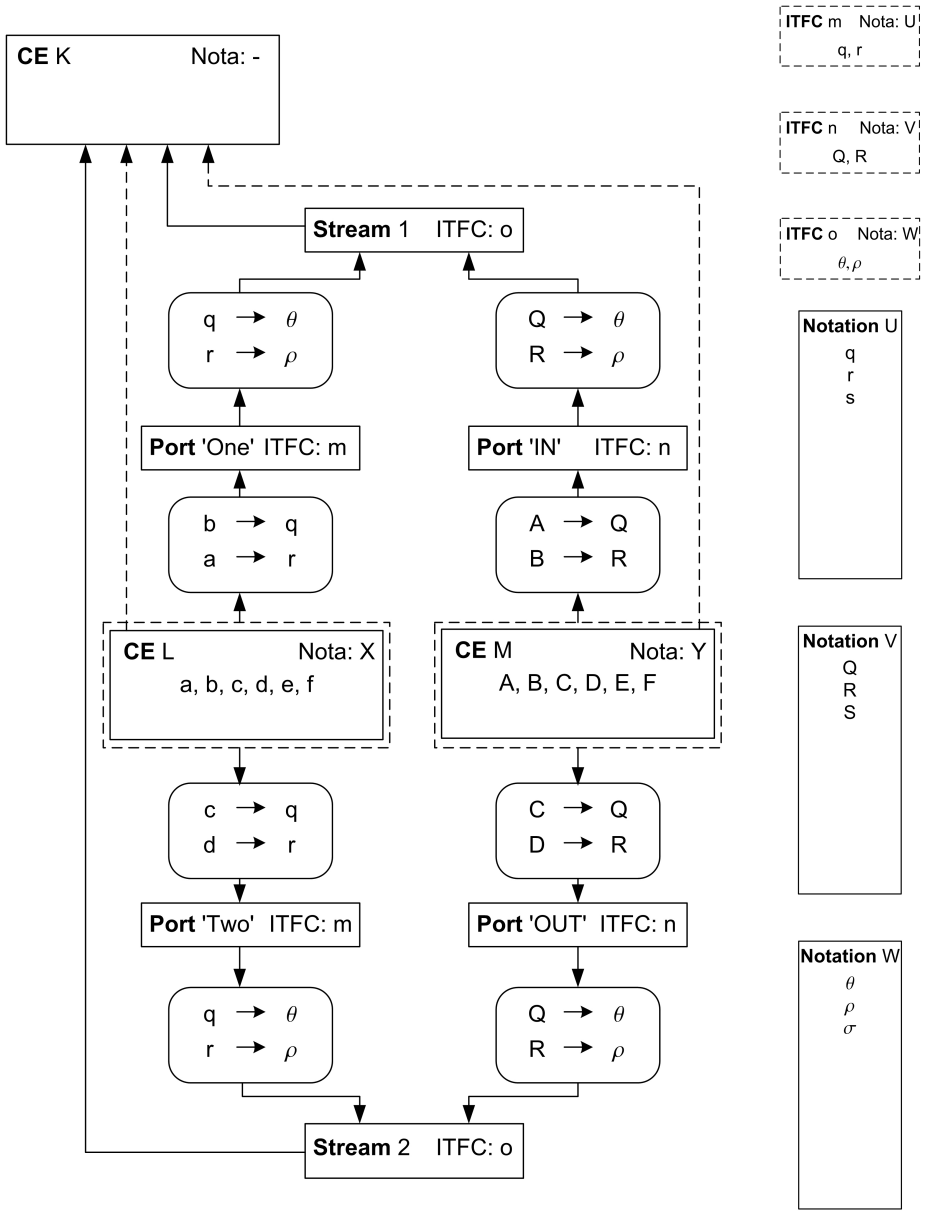


Figure 12: Connection using ports and streams without abbreviated symbols.

3 Mathematical Expressions in MOSAIC

3.1 Conventions for Mathematical Expressions

1. **Introduction**

   This section contains design specifications for mathematical expressions used in MOSAIC. These design rules are independent from the way the expressions are entered (MathML, Tex, Formula editor, etc.). They are laid down as an attempt to

   - avoid mistakable statements, and to
   - limit the complexity of possible mathematical expressions.

   In this text the word 'identifier' refers to any letter that is used to name a variable within a formula.

   Example The formula

   $$a_{j=1}^{L} = 3 \cdot B_{k=2}^{M} \tag{1}$$

   contains the variables $a_{j=1}^{L}$ and $B_{k=2}^{M}$ which are build up by the identifiers a, L, j, B, M, and k.

   The term 'two-dimensional symbolic language' refers to the form of mathematic expressions as used in (1), i.e. using superscripted and subscripted identifiers for the naming of variables and graphic symbols for operations, such as $\sqrt{a}$ for the square root, etc.

2. **Mandatory Description for Identifiers**

   All identifiers and, if applicable, all modular parts of identifiers must be given a description in a notation document.

   Example If there is a Variable Naming

   $$h_{i}^{L} \tag{2}$$

   there must be a description (in a Notation) for all three particles h, i, and L.

3. **Naming of Variables**

   MOSAIC recognizes Variable Names in two-dimensional symbolic language (see above). The names must adhere to the following rules.

   (a) Basic Elements
   A variable has

- exactly one name at the bottom line (base name)
- none or one fixed subscript
- none, one or several fixed superscripts
- none, one or several valued indices, that are as a rule subscripted

In this context 'valued indices' means subscripted language elements that take a value or a range of values, e. g. $(1..NC)$.

(b) Position and separation of sub or superscripted elements

- Several subscripted or superscripted elements occurring in one Variable Naming have to be separated by commas.
- The fixed subscript - if any - takes the first position in the subscript expression.
- The classification of the first subscript into 'fixed subscript' or 'valued index' must be done by an applicable and unambiguous notation.

(c) Fixed setting and offsets for valued indices

- Valued indices might be given a fixed value initially or they are instantiated with a value during evaluation of the equation. In either case this instantiation is expressed by an equality sign.
  e.g. $x_{j,i=1}$
- Indices may be specified by a relative offset (such as j-1 or i+1). The offset must be a natural number. The only valid operations for the expression of valued index offsets are subtraction and addition.
  e.g. V_{j+1}
  Valued indices may be instantiated with expressions that incorporate their maximum value symbol (for $i \in [1..Nc]$, the maximum value symbol is Nc). If an index is instantiated with its maximum value, then it is possible to give an offset of this instantiation value.
  e.g. $x_{i=Nc-1}$

Example In the Variable Naming

$$p_{o,i}^{LV}$$

- There is one fixed superscript: LV ('LV' is seen as one superscript since L and V are not separated by a comma).

- There is one fixed subscript: o (the classification of 'o' as a fixed subscript must be done by an applicable notation element because, e.g. in the above expression 'o' could also be a valued index).
- There is one index: i (which can later take a value, e.g. $i = 1$)

4. **Operations**

(a) Mandatory Expression of Operators

All operators must be expressed. (Thus, the multiplication operator must not be omitted).

Example To express 'A times B is equal to C', the expression $A \cdot B = C$ is allowed. $A\ B = C$ is forbidden. Note that the latter expression could easily be misunderstood as 'variable AB is equal to variable C'.

(b) Mandatory Brackets for Power

Any raising to a power by superscription of variables must use brackets for the base of the power expression. This is to avoid confusion with fixed superscripts. (Keep in mind that fixed superscripts may not be present in the original version of an equation but may be added during variable translation).

Example To express 'variable x^R raised to the power of n', the expression $(x^R)^n$ must be used. The expression x^{R^n} is not allowed because it would lead to confusion with 'variable x raised to the power of R^n'.

(c) Supported Operations

The following list contains the operations that are currently supported by MOSAIC. The operations can be combined and nested just as they can in real mathematical expressions.

Description	Symbols
Basic	$a = b$ $a + b$ $a - b$ a/b $\frac{a}{b}$ $a \cdot b$
Advanced	$(a)^b$ $\exp(a)$ $\ln(a)$
Trigonometric	$\sin(a)$ $\cos(a)$
Summation	$\sum_{i=1}^{Nc} x_i$
Differential	$\frac{d\,a}{d\,t}$

3.2 Latex Standard for MOSAIC

1. **Introduction**

   This section specifies the functionality of the simple Tex to MathML translator. This translator takes MosaicLatex as input and gives MosaicMathML as output. Where MosaicLatex is explained in this section and MosaicMathML is a special form of valid Presentation MathML and is specified in a separate document.

2. **Specification of MosaicLatex**

   MosaicLatex is a sub-set of the Latex Language. To restrict complexity, additional design rules have been added. The following section describes this Latex derivative.

   Any non-compliance to the following rules shall be reported with an error message by the translating software.

   (a) Basic Rules

   - The command set accepted by MOSAIC is limited to commands specified in this document in section 'Commands available'.
   - When using subscripts or superscripts curly brackets are mandatory no matter if there is only one displaced character.
   - All operators have to be expressed. Specifically, the multiplication operator must be present. Missing operators lead to an error message in the translating software.
   - The order of `^{}` and `_{}` is arbitrary. Both variants lead to the same mathematical meaning.
   - In addition to the tex typing rules, the 'MOSAIC math expression convention' (especially the rules for subscripts and superscripts) must be complied with.
   - The whole expression given to the translator is evaluated as one line. Characters that separate the expression into multiple lines are ignored by the translator.
   - The white spaces are ignored in the translator just as they would be ignored in any tex translator.
     Exception (due to the math expression convention):
     A white space between two names leads to an error as it is not clear whether the white space served as an avoided (or invisible) multiplication sign.
     Example: `A=B C` would lead to an error and `A=BC` would be accepted (but translated simply as equality between variables A and BC).

(b) Summation

- `\limits` can be written or not.
- `_{} ^{}`, order of lower and upper bound is arbitrary .
- Curly brackets in lower and upper bounds are mandatory.
- Lower bound consists of identifier, equals sign, number or identifier, e. g. `_{k=i}` or `_{k=1}`
- Upper bound is represented by a number or an identifier: `^{N}` or `^{8}`
- The abstract summand, i. e. the term influenced by the summation indices must be enclosed in curly brackets, e. g. `{a_{i}\cdot x_{i}}`.

Example

```
\sum_{j=1}^{NC}{a_{i}\cdot x_{i}}
\sum\limits_{j=1}^{NC} { a_{i}\cdot x_{i} }
```

(c) Commands available

`\cdot`, `\frac{}{}`, `\sin()`, `\cos()`, `\exp()`, `\ln()`, `\sum`, `\limits`, `\sqrt{}`

(d) Special commands

- Differential operator
  `\diff{a}{b}` stands for the differential of variable a with respect to variable b. **Attention:** This is not a standard latex command. To use this command in documentation it is necessary to define the `\diff` environment as follows:
  `\newcommand{\diff}[2]{\frac{d #1}{d #2}}`

3.3 The correct use of indices

3.3.1 Index specifications in Equations

$$F_j \cdot z_{j,i} = V_j \cdot y_{j,i} - L_j \cdot x_{j,i} + L_{j-1} \cdot x_{j-1,i} + V_{j+1} \cdot y_{j+1,i} \tag{3}$$

[Basic rule:] The symbols for the maximum values must be specified in the Notation.

In the case above the symbols for the maximum values are $i \rightarrow NC$, $j \rightarrow NS$.

Type	Example	Classifications	Model Element
Full[1]	V_j, $z_{j,i}$	generic	Equation
Full w/ offset	L_{j-1}, $x_{j-1,i}$	generic	Equation
Direct w/ value	$x_{j=3,i=5}$	inst	Equation
Direct w/ MaxVal (1)	$x_{j=NS,i=5}$, $x_{j,i=NC}$	inst	Equation
Direct w/ MaxVal (2)	$L_{j=NS-1}$	inst	Equation
All elements explicitly	$i = 1..NC$	generic	Functions, Func Appl

3.3.2 Index specifications in Functions

Functions The following rules apply

1. No indices are allowed in the function interface.
2. The only exception is the use of the specification 'All elements explicitly' (e.g. $i = 1..NC$).

Parameter Lists in Functions Parameters are often used for several components and thus generally carry at least a component index. To make Parameter Lists reusable the component index of such lists and the possible component index in later applications of the function must be separated from each other. The the Parameter Index must be selected and is a property of the function.

[For further development only: A better solution would be taking the Parameter List specification completely out of the Function and make it a property of the Function Appliance instead. Thus the index matching would be simplified down to one single procedure. (How could a downward compatibility established in this case?)]

Function Appliances It is necessary to provide an index matching specification because the Variable Namings concerned may contain more than one index. The applied variable namings may be specified using 'Full' or

[1] Can also be referred to as 'All elements implicitly'

'All elements explicitly'. Please note that no meaning of indices can be anticipated a priori, i.e. for example i cannot be expected to be a component index etc. Such information is subject to the Notation. Examples:

interface	applied	index matching [interface→applied]
$x_{i=1..NC}$	$y^a_{k,j}$ or $y^a_{k=1..NK,j=1..NJ}$	$i \rightarrow k$
$y_{i=1..NC,j=1..NS}$	$z_{i,j}$ or $z_{i=1..NStage,j=1..NComp}$	$i \rightarrow j$, $j \rightarrow i$

[For further development only: Parameter Lists as property, see above]

3.3.3 Index specifications in Connectors

3.3.4 Index specifications in Interfaces

3.3.5 Index specifications in Parameter Lists

3.3.6 Index specifications in Summation expressions

Currently the values for the indices may only run from '1' to the value of the maximum value symbol (e.g. 'NC').

[For further development only: Possible improvements / extensions include:]

- Offsets in maximum value of the sum: $NC - 1$ or $NC + 4$
- Number as maximum value: $\sum_{i=1}^{8}$ (that might restrict the reuse).
- Start value different from one: $i = 3$
- Start value is equal to value of another index: $\sum_{i=j}^{Ni}$

  Observations: it would be necessary to check (1) if j is really used in another summation expression, which might (2) be demanded to be enclosing the one using $i = j$. (3) The values given to the Max Vals of both indices should be the same.
- Start value is not equal to value of another index: $\sum_{i \neq j}^{Ni}$.

  Observations see above.

3.3.7 Index specifications in Evaluation::Indexing Panel

Currently the following rules apply:

1. It is only possible to specify an *integer value* the maximum value symbol (e.g. NC).

2. The counting of the indices always starts at the value '1'.

[For further development only:]

- Does it make sense to allow a different specification of the minimum value (e.g. '0' or '-3'?)
- Does it make sense to allow complex specifications like $j \rightarrow$'$1..5, 7..10, 12..NS$'
- (Both above suggestions possibly restrict the reusability, but do they have merits anyway?)

3.4 Index values as Variables

[For further development only: Both the values of indices and the Maximum Value Symbols should be available as Variable. The resulting restrictions include the following:]

- The variables that really are indices (1) must not be part of the variable list, (2) they should be replaced by their actual value during instantiation (which is possible, since the maximum value is known at this point).
- Notations should not use identical Strings for 'Base Names' and 'Indices' (example: if k is used as an 'Index' it should not be listed as possible 'Base Name'). The creation of such ambiguous Notations should be prevented by the GUI. For downward compatibility: If (1) ambiguous Notations are encountered during EQ-System parsing *and* (2) an equation contains an ambiguous symbol as variable, it should be asked if the symbol is a Base Name or an Index.

4 Standard Components of the User Interface

4.1 File Selection

File Panel Here the user can open files within the user's account on the server. [`Open`], [`Save`], and [`Save As`] allow the corresponding file operations by the help of the file selection dialog. [`New`] clears the section of the user interface that contains data for this file panel.

Figure 13: File Panel

Load File Panel Allows the user to load model files where there is no need to modify them or where further modification is not wanted. [`Change`] opens a file selection dialog window, [`Reload`] is useful if the loaded file has been changed in the meantime in another editor of the modeling environment[2]. [`Unload`] can be used to remove model elements completely from the implementing element (e.g. to remove a Parameter List from the Equation).

Figure 14: Two Load File Panels, one for Notations and one for Parameters. The latter allows the user to eventually unload files if necessary.

4.2 Tables

Multiple Editing To facilitate the process of entering numbers into tables, it is possible to enter the same value into several rows. To use this functionality, select the rows to be edited and click with the right mouse button into one of the selected cells.

[2] In MOSAIC, all editors are independent from each other. That means, e.g. that within the `Equation Editor` it is not recognized if a Notation is loaded or changed `Notation Editor`. If a Notation has been changed in the `Notation Editor` and the changes are needed in the Equation currently opened in the `Equation Editor`, then the Notation must be [`Reload`]ed.

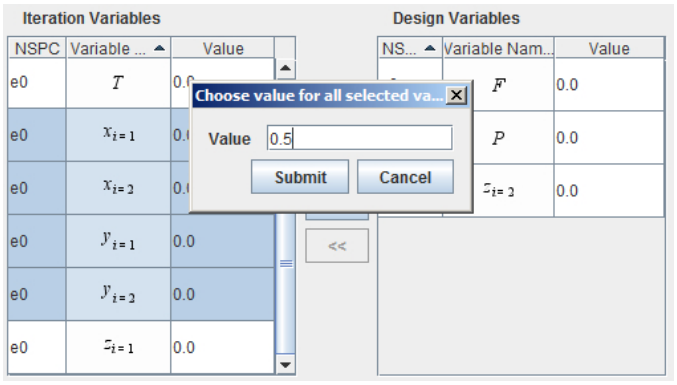


Figure 15: Multiple selection and editing of cells

4.3 Math Editors

Equations and Functions These user interface elements allow the creation of `MathML` code. They consist of a text area that allows the user to enter or modify the MosaicLatex expression and a MathML display area. The creation of the MosaicMathML expression is not done automatically. It must be called for by pressing [`Generate MathML`]. If the MathML expression is up to date and if a notation is assigned to the expression, it can be tested if all variables in the parsed MathML expression are described by the notation. This can be done by pressing [`Test Nota Compliance`].

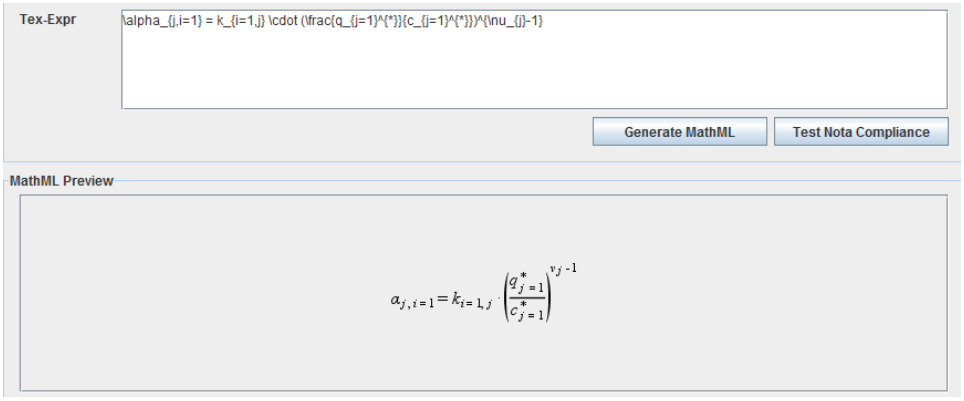


Figure 16: Variable Naming Dialog

Variable Naming Editor Allows the specification of Variable Namings by indicating a MosaicLatex expression. Please do not forget to use the [`Generate MathML`] after a MosaicLatex expression was entered or modified.

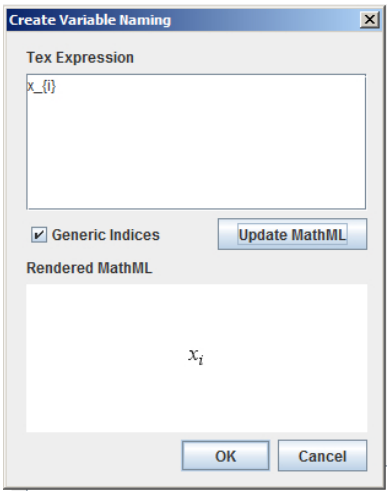


Figure 17: Variable Naming Editor

4.4 Modeling assistance

Variable Naming Details Panel Shows notation information for all elements of the Variable Namings belonging to a selected Variable. If a Variable has several Variable Namings it is possible to scroll through them by using the arrow buttons. Pressing [T] brings up the Top Level Naming of the Variable.

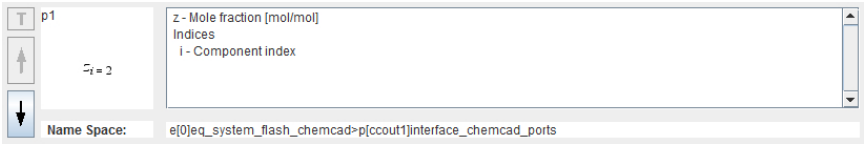


Figure 18: Variable Naming Details Panel

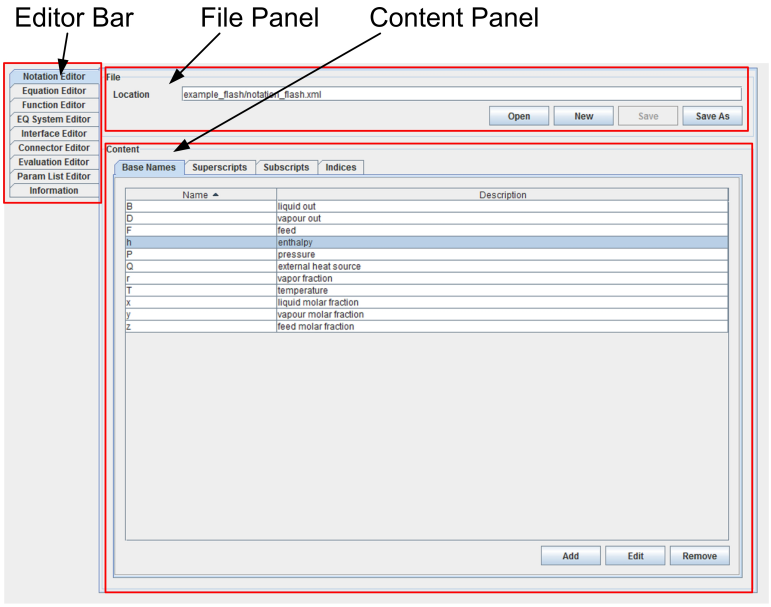
5 Model parts and corresponding editors

In this section all modeling parts in MOSAIC are introduced and their creation and use is explained. For more information on concepts and applications use the Tutorials and Examples.

Structure of the tool The MOSAIC user interface consists of editors for the different modeling objects. You can switch between these editors in the Editor Bar. The editors have the same basic structure: They consist of a File Panel providing functionalities for loading, saving and creation of files,

and a Content Panel that allows the viewing and editing of the content of the files.

Note that each editor has its own File Panel. You can load, manipulate and save different objects independently. However, only *saved* information is considered in the model.



5.1 Notation Editor

To create or manipulate Notations, choose the `Notation Editor` on the Editor Bar. Now you can use the notation editor's File Panel to open an existing Notation, save changes or create new Notations.

In the Content Panel, to add a base name chose the tab `Base Names` and click on [`Add`]. In the upcoming dialog window enter the symbol and the description. You have to give a description for every symbol you enter. You can use the buttons [`Edit`] and [`Remove`] to modify your list of symbols. If you double click on an entry, the `Edit` dialog for this entry is shown.

Indices are different from the other symbols: you have to specify a symbol for the maximum value, which should be a symbol and not a concrete number. Example: To provide an index for the components of a mixture you might specify Name: 'i', Max. value: 'NC', Description: 'Component index'

Note: Please make sure that you save all changes of your notation because MOSAIC will use it as it appears in the corresponding file, and unsaved changes are not considered.

5.2 Equation Editor

To create or manipulate Equations, choose `Equation Editor` on the Editor Bar. Now you can use the equation editor's File Panel to open an existing equation, save changes or create new equations.

In the Content Panel you see the fields

- `Notation` - Here you specify the notation that is used to identify the meaning of the symbols used in the equation. You can choose a notation using the Load File Panel.

  This field is mandatory.

- `Parameters` - If your equation contains parameters that appear repeatedly in the whole model you can separate them here from the other variables by loading a Parameter List. See section **2.2.8 Parameter Lists** for more information.

  5.5.6 Parameter Specification describes how to define parameter lists.

  The values for the parameters are set in the Evaluation. See **5.5 Evaluations**.

  This field is optional.

- `Description` - Here you have to give a description for the equation.

  This field is mandatory.

- `Tex-Expr` - Here you can create the mathematical expression using latex. For more information on the MosaicLatex formulations recognized by MOSAIC see section **3.2 Latex Standard for MOSAIC**.

  This field is mandatory.

Once you have specified a notation and entered a suitable MosaicLatex code, press [`Generate MathML`]. If the latex code could be resolved by MOSAIC you will see a corresponding rendered mathematical expression in the `MathML Preview` area.

To be able to use this equation in an equation system, all symbols in the expression must be specified by the selected notation. To test if the expression is compliant with the currently selected notation press [`Test Nota Compliance`]

Note: Make sure that you save the changes to the loaded equation before you use it in any equation systems.

5.3 Function Editor

To create a new function or manipulate an existing one, choose `Function Editor` on the **Editor Bar**. Now you can use the function editor's File Panel to open an existing function, save changes or create new functions.

In the Content Panel you see the fields:

- `Notation` - Here you specify the notation that is used to identify the meaning of the symbols used in the equation. You can choose a notation using the Load File Panel.

  This field is mandatory.

- `Parameters` - If your function contains parameters that appear many times in the whole model you can separate them here from the other variables by loading a Parameter List. See paragraph 'Parameters' in **5.2 Equation Editor**.

  This field is optional.

- `Description` - Here you must give a description for the equation.

  This field is mandatory.

After you have loaded a notation you can specify the function itself. In the lower section of the Content Panel you will see four tabs:

- `Output Value` - Here you can enter or modify a Variable Naming for the output value.

- `Input Values` - Here you can specify a list of Variable Namings representing the input values of the function.

- `Param Set Index` - If you use a parameter list (see **2.2.8 Parameter Lists**) you can specify an index of this list to be used in the function.

- `Formula Expression` - Here you specify the mathematical expression for the calculation of the output value. See section **3 Mathematical Expressions in MOSAIC**.

5.4 EQSystem Editor

To create or manipulate Equation Systems, choose `EQSystem Editor` on the Editor Bar. Now you can use the EQSystem Editor's File Panel to open an existing equation system, save changes or create new equation systems. It is *recommended* that you read section **2.3 Connection Techniques** before you continue here.

In the Content Panel there is a field `Notation`. You must specify a notation here (using the Load File Panel). Below you see four tabs:

- `Connected Elements` - here you can add equations and other equation systems to your equation system.
- `Internal Streams` - under this tab you can connect several equation systems by streams.
- `External Ports` - here you can add external ports to your Equation System.
- `Functions` - use this tab to add the application of functions to your equation system.

5.4.1 Connected Elements Editor

This dialog allows you to add 'Connected Elements' (other equations and equation systems) to the equation system you are editing. The following control elements are present:

- A `Conn.Elem.` field where you load the XML-file for the connected element from the file system by clicking [`Change`]. The dialog loads the connected element and determines its name and type. It also checks whether the connected element is an equation system containing ports.
- A drop down list `Naming policy` allows you to decide how to transfer the variables of the added connected element to the equation system you are editing.
- A check box `Use connector` and a related file system access field. If you decide to use a connector to find suitable synonymous Variable Namings for the new variables you can load a Connector element here.
- A check box `Apply external indices` and a related editable list.

5.4.2 Internal Streams Editor

This dialog allows you to create or edit the connection of equation systems containing ports. The prerequisite to connect two equation systems with this dialog is that you have added them to your equation system before by using the Connected Elements Editor. The dialog has the following elements:

- A panel `Relation` containing a file loading field `Interface`. Here you specify the Variable Namings that will be used in the stream. The variables in both connected ports have to be transfered to these Namnigs by the connectors specified below.
- Two identical panels specifying the connection between the stream and each port, `Port One` and `Port Two`. Each of the areas contains

  - A section `Conn Elem` where the equation system belonging to the port is specified by:
    * `Id`, the identification number of the connected equation system in the equation system containing the stream, which is also the equation system that is currently edited.
    * `Name`, the name of the connected equation system, and
    * `Port`, the name of the port of the connected equation system

    If you click [`Change`] in the `Conn Elem` section, a dialog named `Choose Port` is presented to you where you can see the all equation systems that contain ports in the equation system currently edited. If you select one of them, a list of all ports of the selected equation system is given in the lower part of the dialog. For your information the interface used by each port is also indicated. If you select the desired port and click [`Ok`] then all information in the section `Conn Elem` is loaded and set accordingly. You do not need to set these data 'by hand'.

- Below the section `Conn Elem` you find a file loading field `Connector`. Here you specify the connector that will be used to match the Variable Namings in the ports to the Variable Namings in the stream. As explained in greater length in section **2.3 Connection Techniques**, this matching of Variable Namings is between the interfaces of the respective port and the stream.

5.4.3 External Ports Editor

External ports allow you to handle the equation system more like a *unit* in the chemical engineering sense of the word. From a general point of view, adding ports is providing the equation system with standardized outputs. For more information on ports see **2.3 Connection Techniques**. The External Ports Editor has the following control elements:

- a text field `Port Name`, where you can enter the name of the port to be added,

- a load file field `Interface`, where you specify the Interface to be used. By the help of the Interface you define the set of variable names that are used in the port. Thus the interface is the means of standardization here.

- a load file field `Connector`. The connector chosen here determines which variables of the equation system are mapped to the Interface of this port.

5.4.4 Function Usage Editor

Functions are very useful if you need to calculate the value for a variable that is dependent on a finite number of variables in your equation system. This is the case for physical properties but also for many other situations. Specifically, replacing equations with functions reduces the size of the system. The Function Usage Editor is the way to use functions that have been created by the Function Editor (see **5.3 Function Editor**) in an equation system. The input and output Variable Namings in the function itself are called Generic Namings while the corresponding variables in the equation system are call Applied Namings. In the editor you will find the following control elements:

- A file loading field where you can load the function to be used.
- An area named `Preview` which displays the rendered MathML information of the loaded function.
- A field specifying the notation used in the function.
- An area named `Appliances` where you specify how the function's input and output variables will be applied in the equation system. This area is subdivided as follows
  - On the left hand side you find the section for the output variables. A field `Generic Naming` displays the output variable as it is named in the function. A table with only one column named `Applied Namings` contains a list of variable names that are present in the equation system. The variables belonging to those names will be calculated by the function.
  - On the right hand side you see the section of the input variables which contains only one table with the columns `Generic Naming` and `Applied Naming`. For every function appliance you see which variables of the equation system are applied to which variables in the function interface. If in the section of the *output variables* you select one of the `Applied Namings`, the corresponding matching list between `Generic Namings` and `Applied Namings` for the output variables is shown in this table.
  - At the bottom there is a Variable Naming Details Panel that displays the notation information for the Variable Naming that is selected or has been clicked on.
  - Below the tables you see a button row where you can add, edit, or remove appliances. If you press [`Add`] or [`Edit`] a dialog Function Appliance Editor is shown. There you see once again the `Generic Namings` and `Applied Namings` for output and input variables

respectively. The [Edit] button belonging to the output variable on the left hand side allows you to specify the Applied Naming for the the output variable. The [Edit] button belonging to the input variables on the right hand side allows you to specify the Applied Naming for the generic input variable *that is selected* in the table. At the bottom you see a Variable Naming info field.

5.5 Evaluations

To create or manipulate Evaluations, choose `Evaluation Editor` on the Editor Bar. You can now use the evaluation editor's File Panel to open an existing evaluation, save changes or create a new evaluation.

In the Content Panel you find a set of tabs. They are first listed in brief and afterward explained in greater length in individual sections:

- `Equation System` - Here you can load the equation system and see its contents
- `Indexing` - Here you specify the maximum values of all indices present in the loaded equation system
- `Instance` - Here you can see the equation system object that has been created based on the indexing information given in the previous tab.
- `Info` - Here you find information about the name spaces that are present in the equation system
- `Variable Specification` - Here you choose the design variables. Furthermore, you can give values for the design variables and guess values for the iteration variables.
- `Parameter Specification` - Here you specify the parameters that are known globally in the equation system.
- `Evaluation` - Here you generate a code in the programming language of your choice
- `Results` - Here you can have a look at the simulation results

5.5.1 Equation System

In the section belonging to this tab you load the equation system that will be the bases of your simulation problem. It contains the following control elements:

- A Load File Panel where you can load the equation system.

- A tabbed field where you can look at the loaded equation system. The tabbed field contains the tabs `Equations` and `Functions`. Each tab contains a table listing the rendered elements of the equation system and the button [`Row Height`] that allows the user to adjust the height of the table rows in case the formula expressions are very large.

5.5.2 Indexing

In the section belonging to this tab you find a list containing all the indices that have been used in the equation system together with the name for their respective maximum value. The cells in the column `Max Val` can be edited so that you can dimension the equation system according to your actual application. Below the table there is a status information field and a button [`Change`]. This button is used to apply the index values. It is only activated if index maximum value data have changed and if there is a value for all index maximum values.

5.5.3 Instance

In the section belonging to this tab the instantiated equation system is shown. 'Instantiated' means that an equation system object has been created according to the information in the `Indexing` tab. If the equation system does not contain any indices the instantiated equation system looks much the same as the generic equation system in the `Equation System` tab (see above). If there are indices, however, you will note the difference in the number of equations and in the formulation of summations, depending on how the indices have been used. This section contains the following elements:

- a table where the equations are listed,
- a button [`Ports Level`] which allows the user to show the variable names valid at the level of external ports,
- a button [`Name Spaces`] where you can toggle the display of the name spaces belonging to the variables,
- a button [`Row Height`] where you can adjust the row height in the case of very large equations.

5.5.4 Info

The section belonging to this tab contains a table where the name spaces used in the equation system are listed. In the column named `Full Expression` you will see rather long names. These names are distinct and contain information about the connection hierarchy in the equation system. In the

column `Short` you find an abbreviated symbol for this name space. The latter symbol is also displayed in equations and Variable Naming lists.

5.5.5 Variable Specification

In this section you select the design variables, and also give values to design and iteration variables. It contains the following elements:

- a status field `Degree of Freedom` giving the DOF for the actual selection of design variables,
- a table containing the Variable Namings of the `Iteration Variables`. This table has the columns:
  - `NSPC`: Containing the abbreviated name space
  - `Variable Naming`: Containing the Variable Naming without the name space
  - `Value`: Containing the value of the variable
- a table containing the namings of the `Design Variables`,
- a table containing the namings of the `Dependent Variables`, which are variables that are calculated by functions,
- a File Panel that allows you to load and store files on the server. Here you can load variable specification lists or store the list created in the MOSAIC User Interface.
- a Variable Naming Details Panel which allows you to skip through the synonyms of variables (i.e. their different Variable Namings) and show the notation information for the activated naming of the selected variable.

5.5.6 Parameter Specification

This section allows you to give values to the global parameters of the equation system. It contains the following elements:

- a list of the global parameters appearing in the equation system.
- a Load File Panel that allows you to load and store files on the server as well as on the local computer of the user. See section **5.5.5 Variable Specification**.
- a Variable Naming Details Panel where notation information is displayed for the variable that is selected in the list.

5.5.7 Evaluation

The section belonging to this tab is used to generate the code. It contains the following elements: On the tab `Generation`

- A panel named `Status of Information` containing a check list of the steps to be done prior to code generation. If the information is not complete or inconsistent, hints are displayed so that the user can act accordingly.
- A general status information text saying either `Information missing` or `Ready for evaluation`
- A panel named `Language Specification` where you can either activate a drop down list for the selection of `Predefined` language specificators or active a file panel that allows you to load a user defined language specificator.
- A table named `Code Generator and Solver Properties` which displays the adjustable properties for the selected code generator. The values in the table can be changed by clicking in the corresponding cell.
- A button [`Generate Code`] which allows the user to generate program code according to the model information entered in the Evaluation Editor. This button is activated as soon as the model information is complete. If the model information is changed after code generation then the code is marked as outdated. If the changed information is consistent and complete you can bring the code up to date at any time by pressing the button [`Generate Code`] again.
- A button [`Evaluate`] which is activated if code has been generated and if this code is up to date, i. e. if the model information has not changed after code generation.

On the tab `View Code` a text area is provided that allows you to have a look at the code or copy the code for local use.

5.5.8 Results

The section belonging to this tab shows the results of a simulation that has been executed on the server. There are different tabs that are activated according to the type of equation system that has been solved.

5.6 Interfaces

To create or manipulate Interfaces, choose `Interface Editor` on the Editor Bar. Now you can use the interface editor's File Panel to open an existing interface, save changes or create new interfaces.

In the Content Panel you have to load a notation that will specify the symbols used in the interface. After that, you can use the buttons [`Add`] and [`Remove`] to modify the list of Variable Namings. For more information on Variable Namings see section **3 Mathematical Expressions in MOSAIC**.

5.7 Connectors

To create or manipulate Connectors, choose `Connector Editor` on the Editor Bar. Now you can use the connector editor's File Panel to open an existing connector, save changes or create new connectors.

In the Content Panel you find four tabs:

- `View Connector` - This tab gives you an overview of the information stored in the loaded connector object.
- `Set Notations` - Here you specify the notations that are used in the equation systems or equations to be connected. The [`Import`] functionality allows you to select a notation directly or do import it from a connected element or from an interface. The import of notations from other elements has the advantage that the Variable Namings contained in these elements are loaded as suggested Namings for matching.
- `Edit Matching` - Here you can specify pairs of synonymous Variable Namings. The pairs specified in this way are displayed in the list named `Matching`. The list named `Index matching` contains specifications for the connection of generic indices. I If you have imported one or both of the notations from other model elements to be matched, you will already find Variable Namings in the lists named `Subnotation` and `Supernotation`. To define a pair of synonyms, you have to select the variables to be matched in the `Subnotation` and `Supernotation` list. You can only specify one matching pair at a time. Once you have selected one variable in each list, you can define the synonym by clicking [`Match`]. To delete a synonym pair from the `Matching` list, select the pair in question and then click `Break`. If you need to use a Variable Naming that is not contained in the lists `Subnotation` and `Supernotation` you can add this naming to either list by clicking [`New Sub`] or [`New Super`] (see sections **4.3 Variable Naming Editor** and **3 Mathematical Expressions in MOSAIC** for more information). Afterwards, you can use the newly created namings to define the desired pair of synonyms.

Please note that only the list `Matching` is stored in the Connector. The other lists are only provided for editing the Connector.

- `View Notations` - Here you can compare the two notations that are dealt with in the loaded connector.
- `Test Connector` - Here you can load any equation or equation system that uses the 'Subnotation' specified in this connector and you can examine if the contained variables are matched in the desired way.

5.8 Parameter Lists

To create or manipulate Parameter Lists, choose `Param List Editor` on the Editor Bar. Now you can use the parameter list editor's File Panel to open an existing parameter list, save changes or create new parameter lists.

In the Content Panel you need to load a notation first. All Variable Namings added in the list must comply to this notation. You can modify the list by using the [`Add`] and [`Remove`] button. For information on entering Variable Namings please see sections **4.3 Variable Naming Editor** and **3 Mathematical Expressions in MOSAIC**.

6 Tutorials

6.1 Getting Started - Create and solve a very simple model

The Rosenbrock Function is well suited as testing problem. For two variables it has the form

$$f(x,y) = (1-x)^2 + c \cdot (y-x^2)^2 \tag{4}$$

This function will now be used as a learning problem. To do this, the two dimensional optimisation problem in (4) is transfered into a two dimensional root finding problem.

$$g_1(x,y) = a \cdot (1-x) = 0 \tag{5}$$
$$g_2(x,y) = b \cdot (y-x^2) = 0 \tag{6}$$

In the following section, a way to solve this problem in MOSAIC is described. It is assumed that you are already logged in to MOSAIC and see the modeling environment in your browser.

Creating the Notation At first it is necessary to create a notation that contains descriptions for all symbols used for the names of the variables, i. e. a, b, x, and y. To create the notation do the following

- Select the Notation Editor in the Editor Bar on the left hand side.
- Choose the tab `Base Names` and use the [`Add`] button to enter the for letters one by one, giving a description for each of them.
- Click [`Save`] to open a file selection dialog.
- In the dialog press create a new directory or package named 'example_rosenbrock' and change into it afterwards.
- Save the notation (e. g. 'notation_rosenbrock.xml').

Creating the equations

- Make the Equation Editor visible by choosing it in the Editor Bar at the left hand side.
- On the editor you find a field named `Notation` with a short information below saying that no notation is loaded. Click on [`Change`] and select the notation you just created in the upcoming dialog.
- Now you have to enter a text into the field `Description`.
- In the field `Tex-Expr` you enter the mathematic formula for the equation. Please type the following into the field:

```
0=a \cdot (1-x)
```

- Click on [`Generate MathML`] to create MathML code. The rendered MathML expression is shown in the `MathML Preview` area at the buttom of the Equation Editor.
- Save the new equation in the directory you just created for this example.
- Click [`New`] to clear the equation editor.
- Create a second equation, loading the same notation and using the latex code

```
0 = b\cdot ( y - (x)^{2} )
```

Creating the equation system

- Choose the EQ System Editor in the Editor Bar at the left hand side.
- First load the notation for this example in the same way it was done in the Equation Editor.
- Activate the tab `Connected Elements` and click [`Add`].
- In the appearing dialog, click on [`Change`] next to the upmost field and select the first equation for the rosenbrock example.
- Click [`Submit`]. The equation should now be listed in the table.
- Add the second equation in the same way.
- Save the equation system in the directory for this example.

Evaluating the equation system The evaluation of an equation system does two things: Firstly all necessary information is specified so that a complete simulation problem is formulated based on the equation system. Secondly code for the solution of the problem is generated and executed.

- Choose Evaluation Editor in the Editor Bar.
- In the tab `Equation System` click on [`Change`] and select the equation system that has been created in the last steps.
- In this example you can directly go to the tab `Variable Specification`. There you see a list that contains all the variables of the equation system.

- Move the variables a and b over into the list for the `Design Variables` by selecting them with the mouse and using the [>>] button. You will note that the `Degree of Freedom` is automatically updated.
- Set the design variables $a = 50$ and $b = 100$ by clicking into the value field.
- Change to the `Evaluation` tab. If everything is ok, the [`Generate Code`] button should be activated. (Otherwise please have a look at the `Status of Information` in the middle of the editor where you find hints to solve the problem).
- In the drop down list `Code Gen` select the code generator `GSL Hybrid`.
- To create the problem solving code press [`Generate Code`].
- Press [`Evaluate`] to have the code executed on the server.
- Select the tab `Results` to have a look at the solution.

6.2 Notations and Variables - Superscripts, Subscripts, and Indices

This tutorial will demonstrate the use of different elements of a name for a variable. The following simple flash calculation will serve as example. First the model equations are indicated:

Equation System

$$F \cdot z_i = B \cdot x_i + D \cdot y_i \tag{7}$$

$$y_i = K_{o,i}^{LV} \cdot x_i \tag{8}$$

$$1 = \sum_{i=1}^{NC} x_i \tag{9}$$

$$1 = \sum_{i=1}^{NC} y_i \tag{10}$$

$$1 = \sum_{i=1}^{NC} z_i \tag{11}$$

Although the reader may recognize the above equation system as a MESH system without heat balance, it must be pointed out that the physical meaning of the model is not definitely clear before the notation of all symbols is given. Therefore this missing piece of information is provided here:

Notation

- B Bottom molar flow [kmol/h]
- D Head molar flow [kmol/h]
- F Feed molar flow [kmol/h]
- K Constant, phase equilibrium, and others
- x Bottom molar fraction [mol/mol]
- y Head molar fraction [mol/mol]
- z Feed molar fraction [mol/mol]

Superscripts

- LV Phase equilibrium

Subscripts

- o Reference value

Indices

- i Component index, 1..NC

As you will have noticed, the equation system given above has a general character: It describes mixtures with any number of components. Further-

more, the use of the equation system depends on the choice and the value of the design variables. Proving this information is stating a simulation problem. For this tutorial we will state the following problem:

Problem Description

Use the equation system (7) through (11) with

$$NC = 3 \tag{12}$$
$$K_{o,1}^{LV} = 0.8 \tag{13}$$
$$K_{o,2}^{LV} = 0.9 \tag{14}$$
$$K_{o,3}^{LV} = 1.5 \tag{15}$$
$$F = 2\ kmol/h \tag{16}$$
$$z_1 = 0.3\ mol/mol \tag{17}$$
$$z_2 = 0.4\ mol/mol \tag{18}$$

In the following section this simulation problem is transfered into MOSAIC. It is assumed that you are already logged in to MOSAIC and see the modeling environment in your browser.

Creating the Notation At first we need to create a notation. This is done directly as in the written statement under 'Notation' above. To enter the notation do the following:

- Select the Notation Editor in the Editor Bar on the left hand side.
- Choose the tab `Base Names`. In this tab, all symbols that are written in the base line are entered. Use the [`Add`] button to enter the letters B through z using the description from the statement above.
- Now change to the `Superscripts` tab and enter the symbol of the corresponding section above, LV.
- Change to the `Subscritpts` tab and enter the subscripts of the notation above.
- Inter the indices of the notation above in the correponding tab in the Notation Editor.
- Click [`Save`] to open the file selection dialog.
- In the dialog create a new directory or package named 'example_flash' and change into it afterwards.
- Save the notation (e. g. 'notation_flash.xml').

Creating the Equations In this tutorial you will create all equations by yourself. Please note that this step of modeling is not necessary in all cases, as MOSAIC allows and even furthers the use of ready made equations.

The variable names in the given equations contain displacements. How they are expressed in Latex sensible to MOSAIC will be explained briefly here, more information are given in the Mosaic user's guide.

- Superscripts are described using `^{S}` where S is the superscript. In this statements the curly brackets are mandatory. You can have several superscripts. Superscripts are seperated by a comma: `^{S,T}` where S and T are the superscripts. Please note that in the example of this tutorial the combination LV is *one single* superscript.
- Subscripts and indices are described using `_{S}` where S is the subscript or index. Like in superscripts the curly brackets cannot be ommited. You can have up to one subscript and several indices. The fixed subscript must be the first subscripted character. Fixed subscripts and indics are separated by a comma: `_{S,I,J}` where S is a fixed subscript and I and J are indices.

In addition to the upper specifications you will need to know that a multiplication sign is mandatory in MOSAIC. The necessary operators for this example are

- equals: `=`
- plus: `+`
- multiply: `\cdot`

Now you are ready to write the formula expressions yourself while you enter the equations.

- Choose the Equation Editor in the Editor Bar at the left hand side.
- On the editor you find a field named `Notation` with a short information below saying that no notation is loaded. Click on [`Change`] and select the notation you just created.
- Enter a text for equation (7) in the field `Description`.
- Enter a latex expression correponding to equation (7) using the above guide lines.
- If you had problems in entering the formula use the following code

```
F \cdot z_{i} = B \cdot x_{i} + D \cdot y_{i}
```

- Click on [Generate MathML] to create MathML code. The rendered MathML expression is shown in the MathML Preview area at the buttom of the Equation Editor.
- Save the new equation in the directory you just created for this example.
- Click [New] to clear the equation editor.
- Create eqution (8) loading the same notation as in the previous one.
- Again, try to enter the latex code without looking at the code below and press [Generate MathML] to have the latex expression rendered. If the translation fails, an upcoming dialog provides information about the reason of the error. That information should help you to improve the latex expression so that it can be translated. If you closed the dialog window, you can always find the error messages in the tab Information in the Editor Bar on the left hand side.
  If you had trouble entering the equation properly, you may use the following expression:

  ```
  y_{i}=K^{LV}_{o,i}\cdot x_{i}
  ```

- Save the equation in the directory for this example
- Create equations for the summation relation. The necessary latex expressions are shown below. If you like you can use the [Save As] functionality to facilitate your work.

  ```
  1 =\sum_{i=1}^{NC}{x_{i}}
  1 =\sum_{i=1}^{NC}{y_{i}}
  1 =\sum_{i=1}^{NC}{z_{i}}
  ```

Creating the Equation System

- Choose the EQ System Editor in the Editor Bar at the left hand side.
- First load the notation for this example in the same way it was done in the Equation Editor.
- Activate the tab Connected Elements and click [Add].
- In the appearing dialog, click on [Change] next to the upmost field and select the first equation for the flash example.
- Click [Submit]. The equation should be listed in the table now.
- Add the other equations in the same way.
- Save the equation system in the directory for this example.

Evaluation: Using the Equation System to state a Simulation Problem

- Choose Evaluation Editor in the Editor Bar.
- In the tab `Equation System` click on [`Change`] and select the equation system that has been created in the last steps.
- Activate the tab `Indexing`. The table on top of the appearing area should contain exactly one line where the column `Name` contains the value i and the column `Max Gen` contains the value NC. Click into the `Max Val` cell of this line and enter the number '3'. This way you give the **Max**imum **Gen**eric Value NC for i the value '3'. Leave the cell or press Enter.
- Click [`Change`] to apply the new index specification to the equation system. (You can savely answer [`OK`] in the appearing dialog).
- Now choose the tab `Instance` to have a look at the resulting instantiated equation system. Note that there are three equations for (7) and (8) each and that the summation equations have changed too.
- Go to the tab `Variable Specification`. Here you will specify the design variables, bringing the `Degree of Freedom` indicated in the upper left of the operating area to zero.
- Before you proceed, however, click at the variable $K^{LV}_{o,i=2}$ and have a look at the Notation Information Field at the bottom of the user interface. You will notice that for every selected variable listed in one of the tables the information you provided in the notation in the first step of this tutorial is presented here to assist you in your work.
- Select the variable F in the list of `Iteration Variables` and move it to the list of `Design Variables` by clicking [`>>`]
- Move the other variables indicated in the 'Problem Description' above in the same way. Hint: multiple selection is supported like in your operating system.
- Enter the `Value` for F by clicking into the corresponding cell. After you have entered the value (here: 2), leave the cell or press Enter.
- Enter the values for the other design variables in the same way.
- Now you need to specify guess values for the `Iteration Variables`. Hint: you can again make use of the multiple selection functionality. Select all molar fractions, click into the selection with the *right* mouse key and use the appearing dialog to enter the value 0.3 . Select B and D and set them to the value 1.

- Change to the `Evaluation` tab. If everything is ok, the [`Generate Code`] button should be activated. (Otherwise please have a look at the `Status of Information` in the middle of the editor where you find hints to solve the problem).
- In the drop down list `Code Gen` select the code generator `GSL Hybrid`.
- To create the problem solving code press [`Generate Code`].
- Press [`Evaluate`] to have the code executed on the server.
- Select the tab `Results` to have a look at the solution.

6.3 Differential Equation Systems

The following tutorial will show how to describe and solve differential equation systems.

6.3.1 ODE - The Van der Pol Oscillator

In this tutorial the Van der Pol Oscillator will be implemented. The problem is stated as follows:

Equation System Van der Pol

$$\frac{d\,x}{d\,t} = y \tag{19}$$

$$\frac{d\,y}{d\,t} = a \cdot (1 - (x)^2) \cdot y - x \tag{20}$$

Notation Diff Eqs

Base Names:

Name	Description
a	parameter a
t	differential variable t
x	value x
y	value y

Evaluation

- Design value

$$a = 5 \tag{21}$$

- Differential variable and initial values for the state variables

$$t \in [0,\, 20] \tag{22}$$

$$x = 2.0 \tag{23}$$

$$y = 0 \tag{24}$$

Requirements It is assumed that you have basic knowledge about Notations, Equations, and Equation Systems in MOSAIC. If you do not know how to create and modify these model elements, please work through **6.1 Getting Started - Create and solve a very simple model** first.

Creating the Notation

- Create a Notation and enter the names as `Base Names`. If you need help, please follow **Creating the Notation** in tutorial 6.1.
- Note that the differential variable t is part of the notation as well. As far as the Notation is concerned, there is no difference between the identifiers a, t, x, and y.

Creating the equations

- In MosaicLatex there is a special command for the differential operator.

  To create $\frac{dA}{dB}$ please use the code

  ```
  \diff{A}{B}
  ```

- In the current version, MOSAIC only allows first order ordinary differential equations. The differential operator must be on the left hand side.
- Enter the equations one by one as described in **Creating the equations** in tutorial 6.1.
- If you had problems in entering correct code, use the following

  For Equation (19):

  ```
  \diff{x}{t} = y
  ```

  For Equation (20):

  ```
  \diff{y}{t} = a \cdot (1-(x)^{2})\cdot y - x
  ```

Creating the equation system

- The equations can be added to an equation system in the usual way. If you need help, please refer to **Creating the equation system** in tutorial 6.1

Evaluating the equation system

- Load the equation system in the Evaluation Editor and go the tab named `Specification List`. If you need help, see **Evaluating the equation system** in tutorial 6.1

- You will note that x and y are already sorted into the table named `State Variables` while the differential variable t is put correctly into a corresponding section on the right hand side of the user interface.
- Enter the initial values for the state variables and the value for the design variable as stated above in the paragraph **Evaluation** of the problem description.
- Enter the limits of the interval of the differential variable t in the fields named `Start` and `End`.
- Go to the tab named `Evaluation` and make sure that the language specificator `C++ BzzMath ODE Stiff` is activated in `Code Gen` in the `Execute` section. Press [`Generate Code`] and then [`Evaluate`] to solve the equation system on the modeling server.
- Select the tab `Results`. In the now visible section select the tab `DE Variable Values`. Here you find the trajectory list for the differential variable and the state variables.
- Select the tab `DE Plot Results` and tick the boxes for x and y in the column `Plot` of the variables table.

This is the end of the first section of this tutorial. You have learned how to create and evaluate ordinary differential equation systems in MOSAIC.

6.3.2 DAE - The Robertson Problem

In this tutorial the differential algebraic equation system of the Robertson Problem will be solved. The full problem is described as follows:

Equation System Van der Pol

$$\frac{d\,x}{d\,t} = -a \cdot x + b \cdot y \cdot z \tag{25}$$

$$\frac{d\,y}{d\,t} = a \cdot x - b \cdot y \cdot z - c \cdot (y)^2 \tag{26}$$

$$1 = x + y + z \tag{27}$$

Notation Diff Eqs

Base Names:

Name	Description
a	parameter a
b	parameter b
c	parameter c
t	differential variable t
x	value x
y	value y
z	value z

Evaluation

- Design values

$$a = 0.04 \tag{28}$$
$$b = 1e4 \tag{29}$$
$$c = 3e7 \tag{30}$$

- Differential variable and initial values for the state variables

$$t \in [1e-7,\ 1e7] \tag{31}$$
$$x = 1.0 \tag{32}$$
$$y = 0.0 \tag{33}$$
$$z = 1e-3 \tag{34}$$

Notation, equations, and equation system

- Use the same notation as in the Van der Pol example. Only add the missing variables.
- Create the equations and the equation system. Please try to type the equations yourself. If you have problems in entering the code, you may use the following expressions

  For Equation (25):

  ```
  \diff{x}{t}= -a\cdot x + b \cdot y \cdot z
  ```

  For Equation (26):

  ```
  \diff{y}{t}= a\cdot x - b\cdot y\cdot z - c\cdot (y)^{2}
  ```

  For Equation (27):

  ```
  1=x+y+z
  ```

Evaluation

- Load the equation system into the `Evaluation Editor` and select the tab `Variable Specification`
- Declare z as a state variable by moving it to the table named `State Variables` (to do so, select z in the table `Disign Variables` and press [<<]).
- Specify the values for the design variables and the state variables in the corresponding table cells.
- Specify $1e-7$ as the start value and $1e7$ as the end value of the differential variable t in the corresponding fields at the right hand side of the user interface.
- Go to the tab named `Evaluation` and make sure that the language specificator `C++ BzzMath DAE Obj` is activated in `Code Gen` in the `Execute` section. Press [`Generate Code`] and then [`Evaluate`] to solve the equation system on the modeling server.
- Select the tab `Results` and in the now visible section select the tab `DE Plot Results` and tick the box for the variable x in the column `Plot` of the variables table.
- On the lower right hand side of the user interface, below `X-Axis Scale` select `Logarithmic`.
- Activate the graph of the variable z by ticking its box in the variables table.
- HINT: If a selected graph does not show after ticking the corresponding box, unselect it and then select it again.
- Activate the graph of the variable y. The values of y are too small to visualize changes in the same graph with x and z without any adjustments. To make the changes of y over t visible in the graph, specify a scaling factor of $1e4$ in the column `Resizing` in the row of y.

Here is the end of this tutorial section. You have learned how to work with differential equations and how to use the plotting functionality of MOSAIC.

7 Appendix

7.1 Connection using streams

7.1.1 Objects

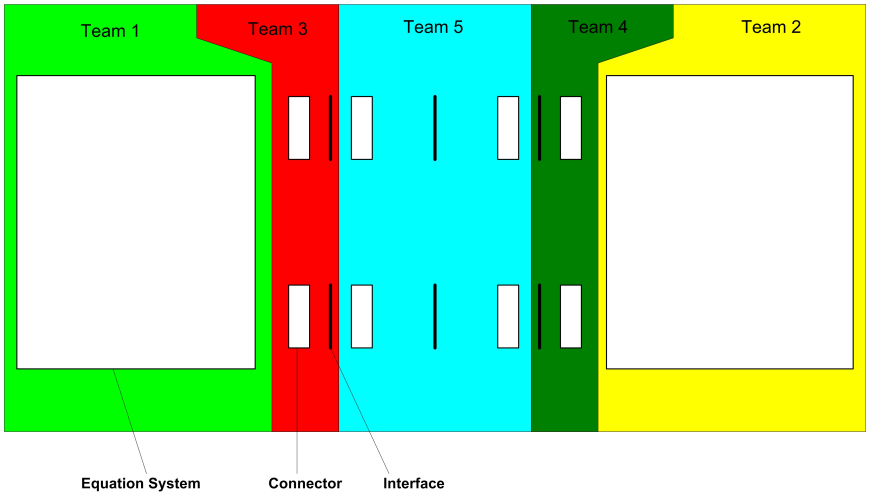
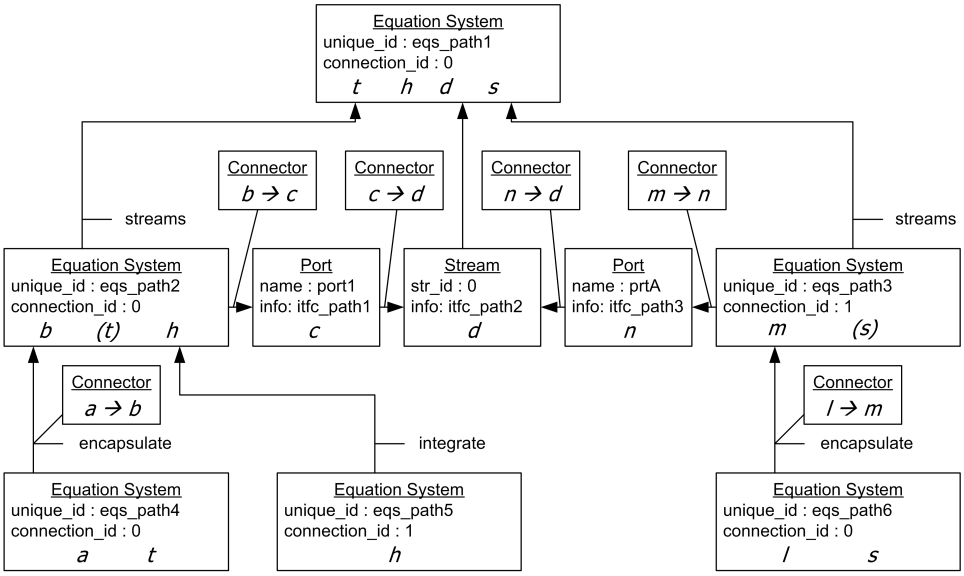


Figure 19: he objects involved in a connection using streams

7.1.2 Name Spaces



Variable 1:

Top Level Naming: *t* in : e[0]eqs_path1>e[0]eqs_path2>e[0]eqs_path4

List of Namings:

t in : e[0]eqs_path1>e[0]eqs_path2>e[0]eqs_path4 → e1

Variable 2:

Top Level Naming: *h* in : e[0]eqs_path1>e[0]eqs_path2

List of Namings:

h in : e[0]eqs_path1>e[0]eqs_path2 → e2

Variable 3:

Top Level Naming: *d* in : e[0]eqs_path1>s[0]itfc_path2

List of Namings:

a in : e[0]eqs_path1>e[0]eqs_path2>e[0]eqs_path4 → e1

b in : e[0]eqs_path1>e[0]eqs_path2 → e2

c in : e[0]eqs_path1>e[0]eqs_path2>p[port1]itfc_path1 → p1

l in : e[0]eqs_path1>e[1]eqs_path3>e[0]eqs_path6 → e3

m in : e[0]eqs_path1>e[1]eqs_path3 → e4

n in : e[0]eqs_path1>e[1]eqs_path3>p[prtA]itfc_path3 → p2

d in : e[0]eqs_path1>s[0]itfc_path2 → s1

Variable 4:

Top Level Naming: *s* in : e[0]eqs_path1>e[1]eqs_path3>e[0]eqs_path6

List of Namings:

h in : e[0]eqs_path1>e[1]eqs_path3>e[0]eqs_path6 → e2